SACRED-MA:
Safe And seCure REmote Direct Memory Access

VeTTS 2024

Guillaume Ambal[1], Gregory Chockler[2], Brijesh Dongol[2],
Milad Ketabi[2], Azalea Raad[1]

[1]Imperial College London, [2]University of Surrey

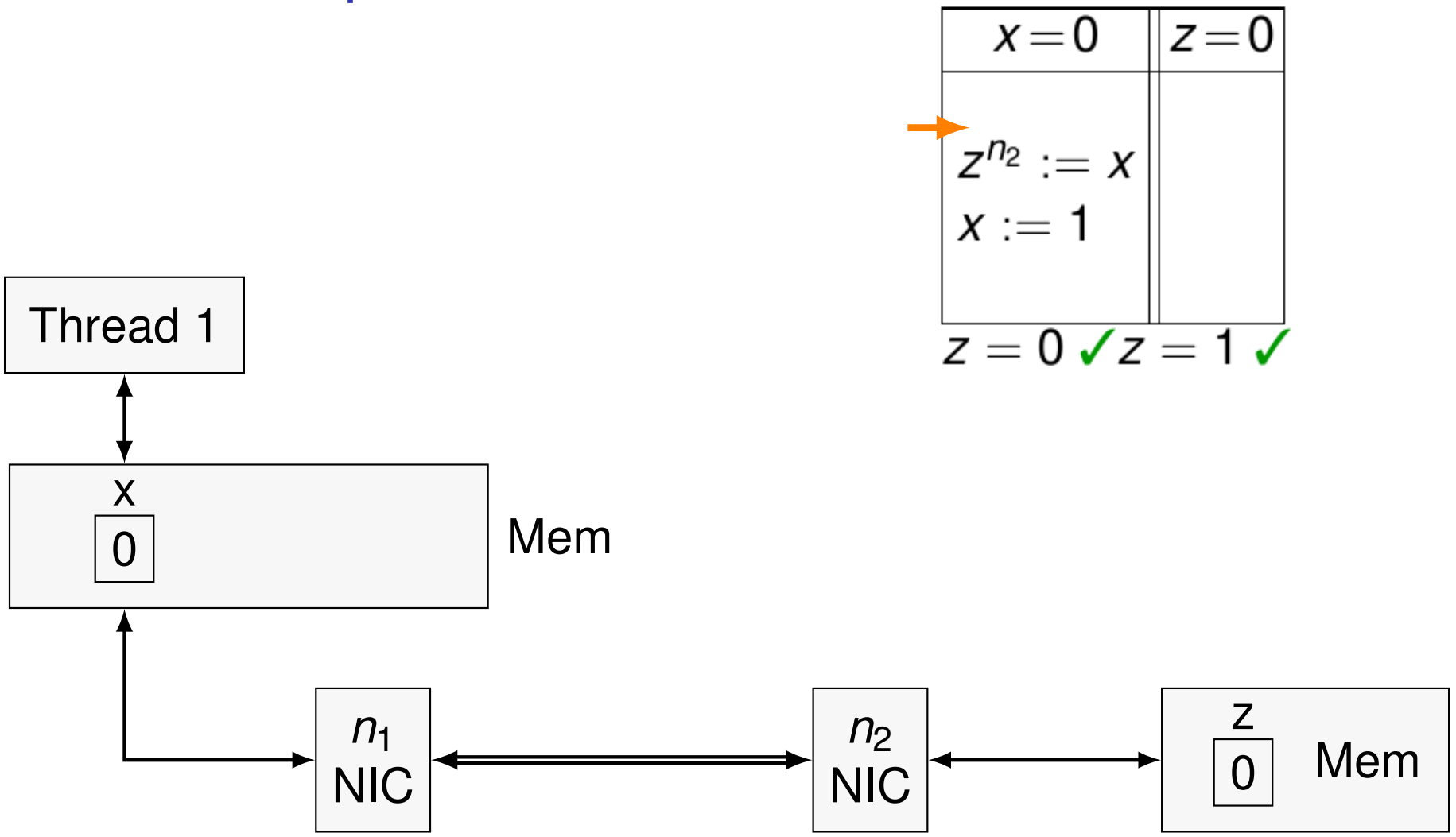
May 21, 2024

RDMA example

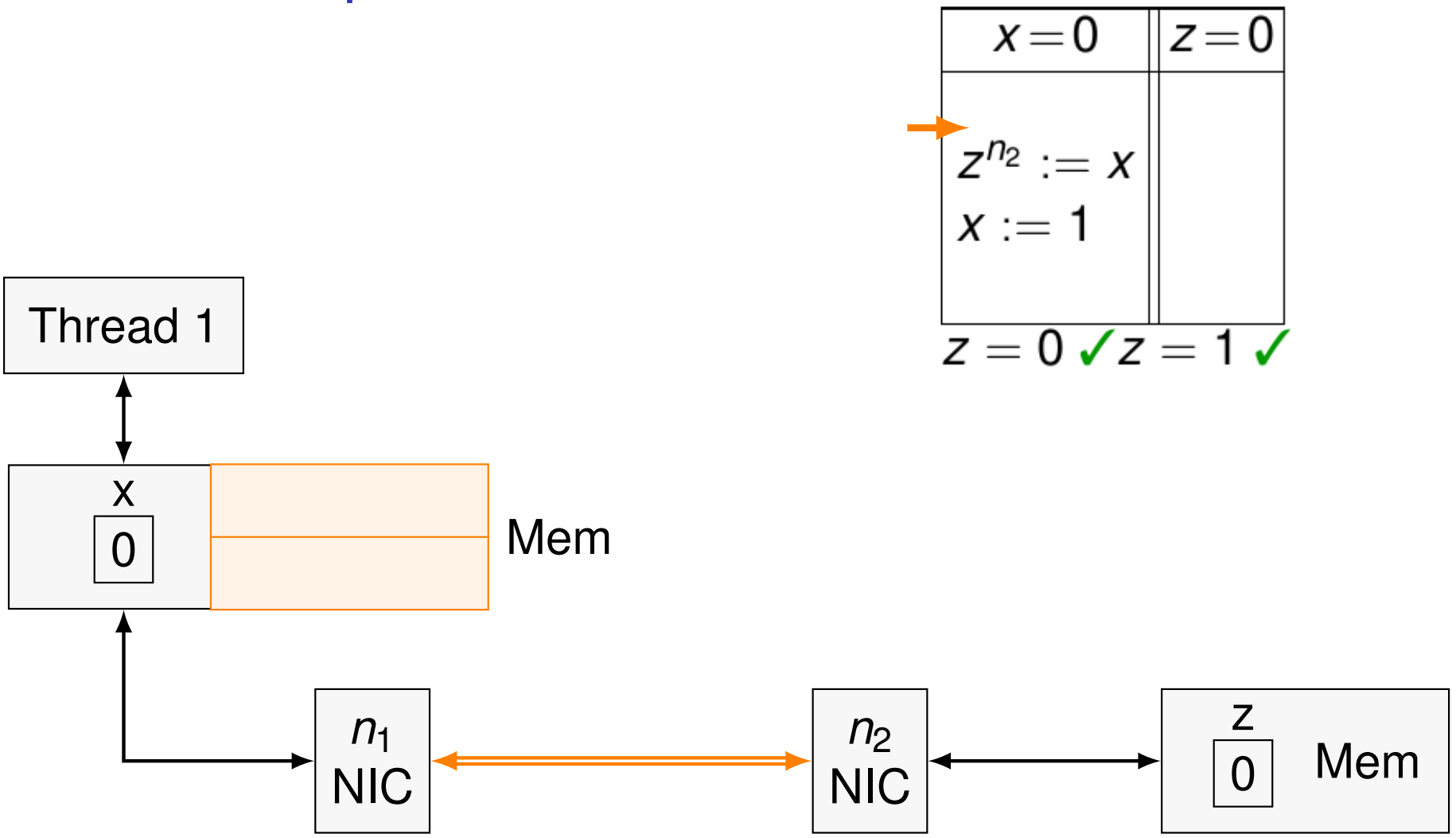
$x=0$	$z=0$
$z^{n_2} := x$ $x := 1$	

$z = 0$ ✓ $z = 1$ ✓

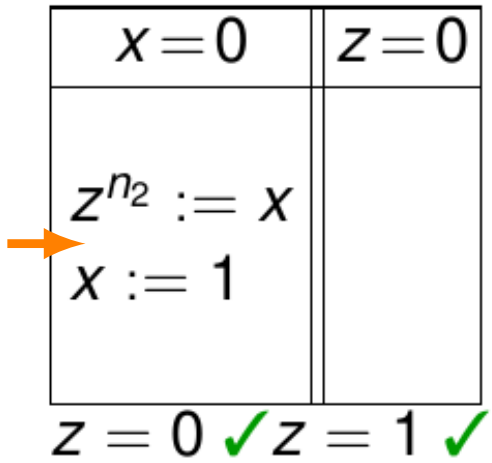
RDMA example

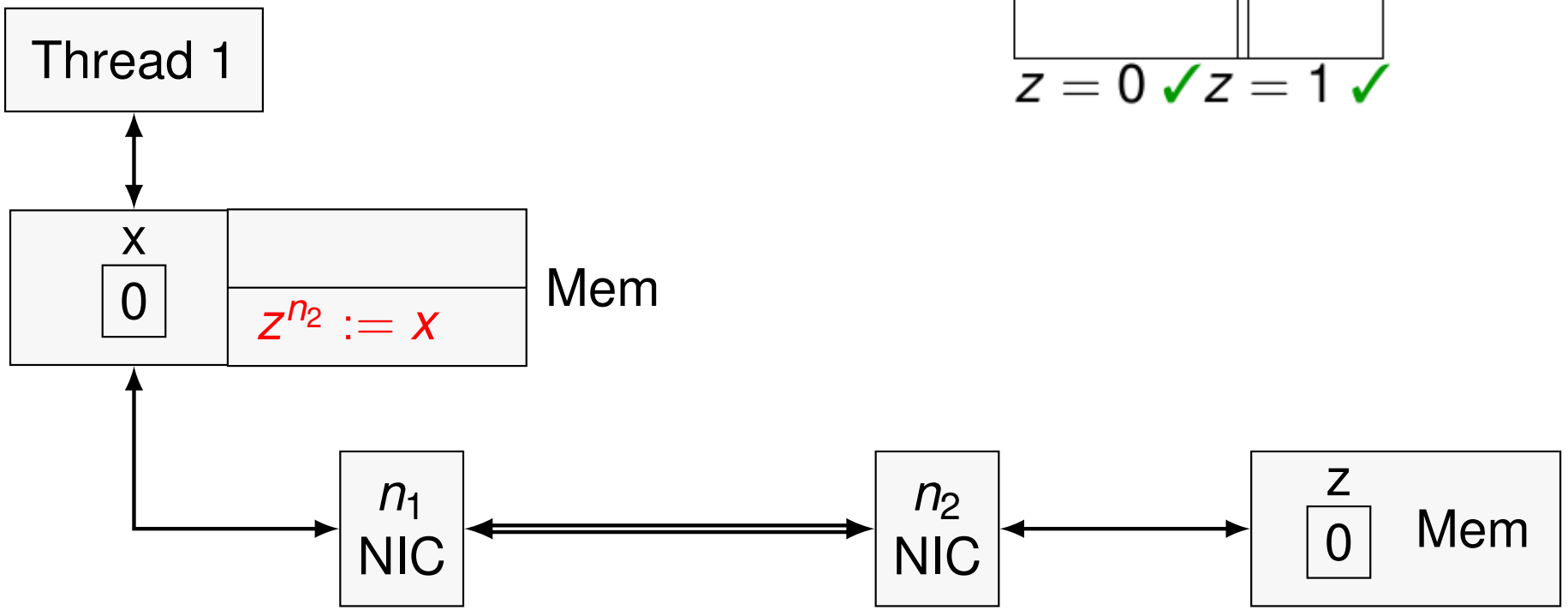


RDMA example

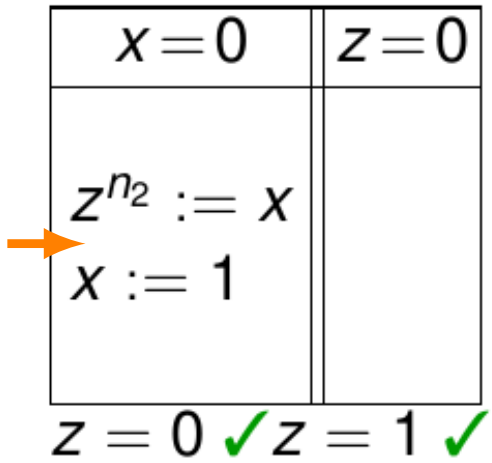


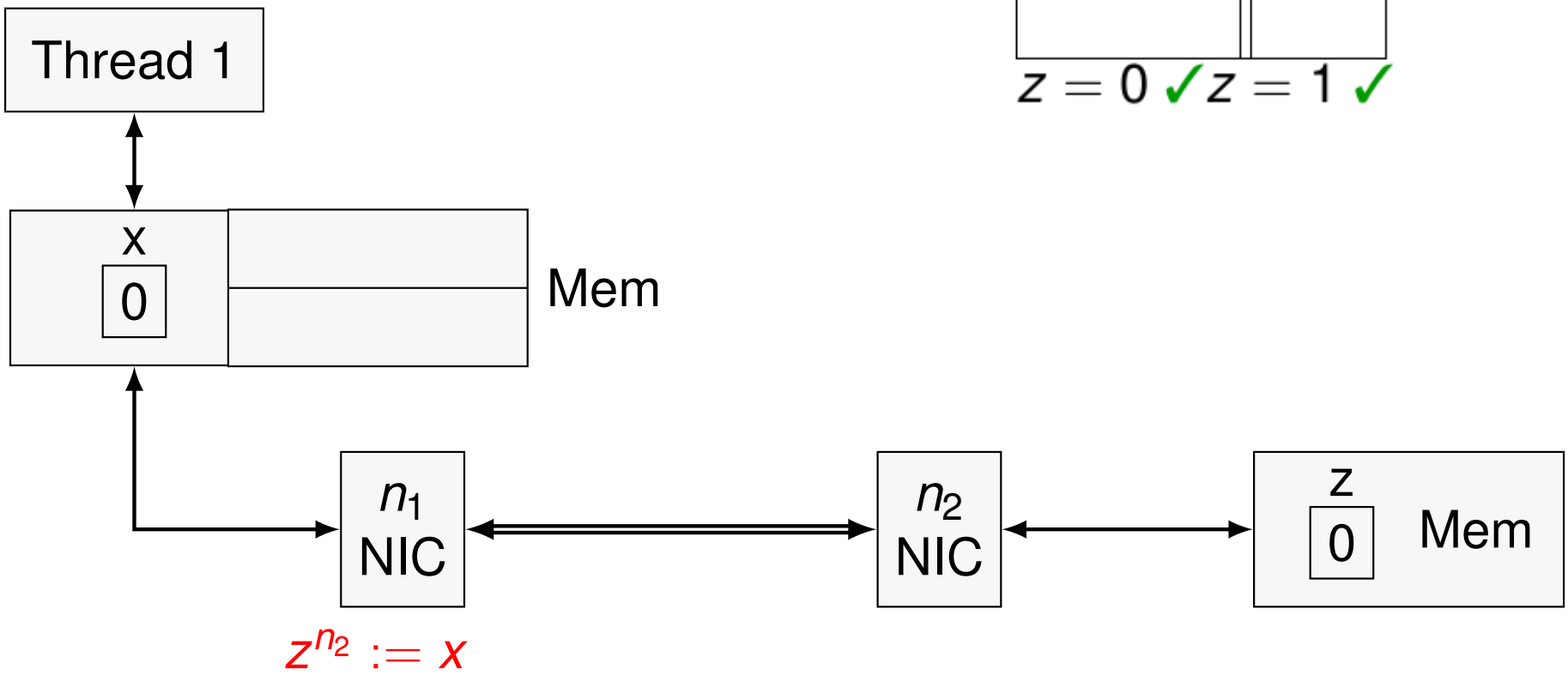
RDMA example



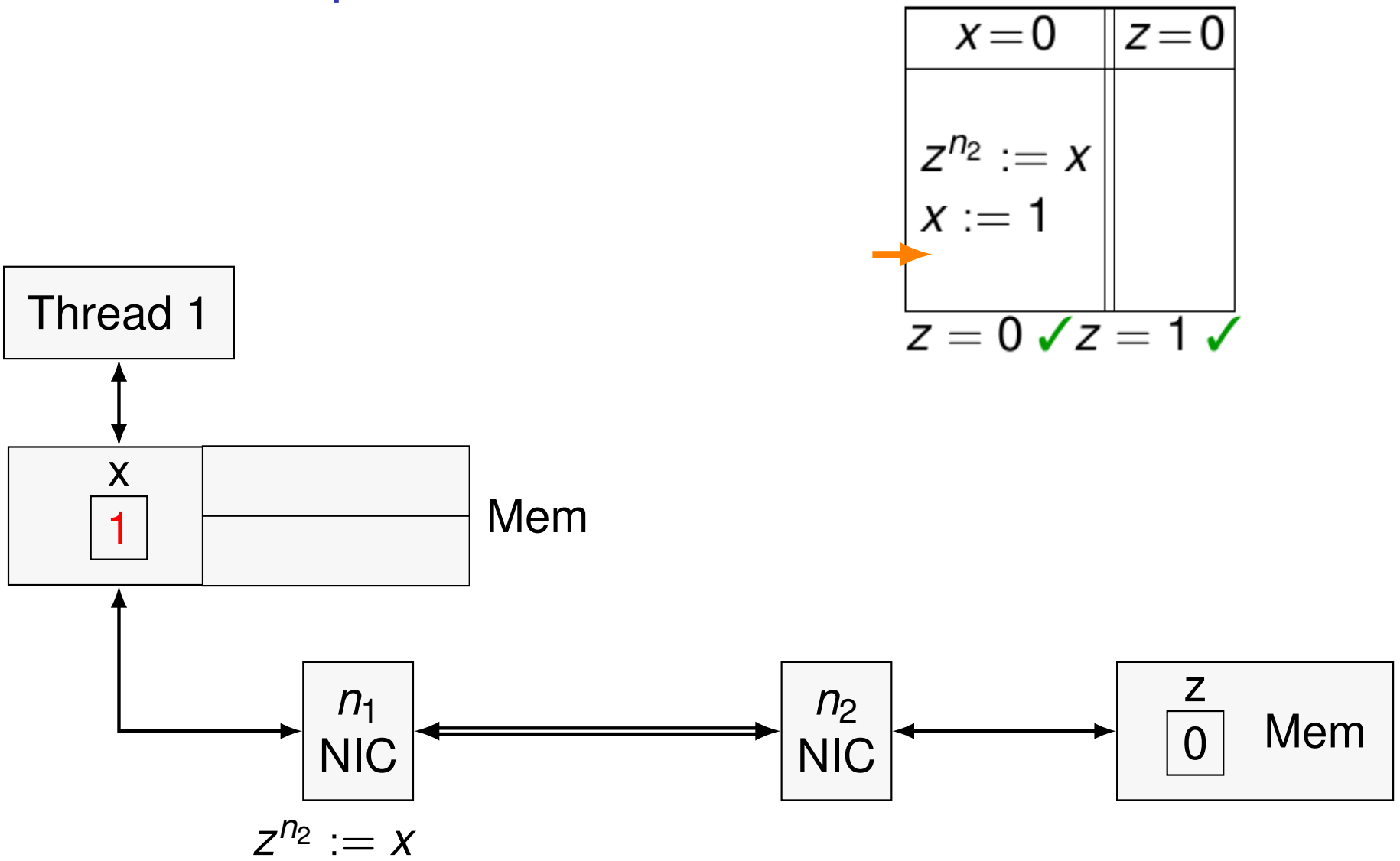


RDMA example

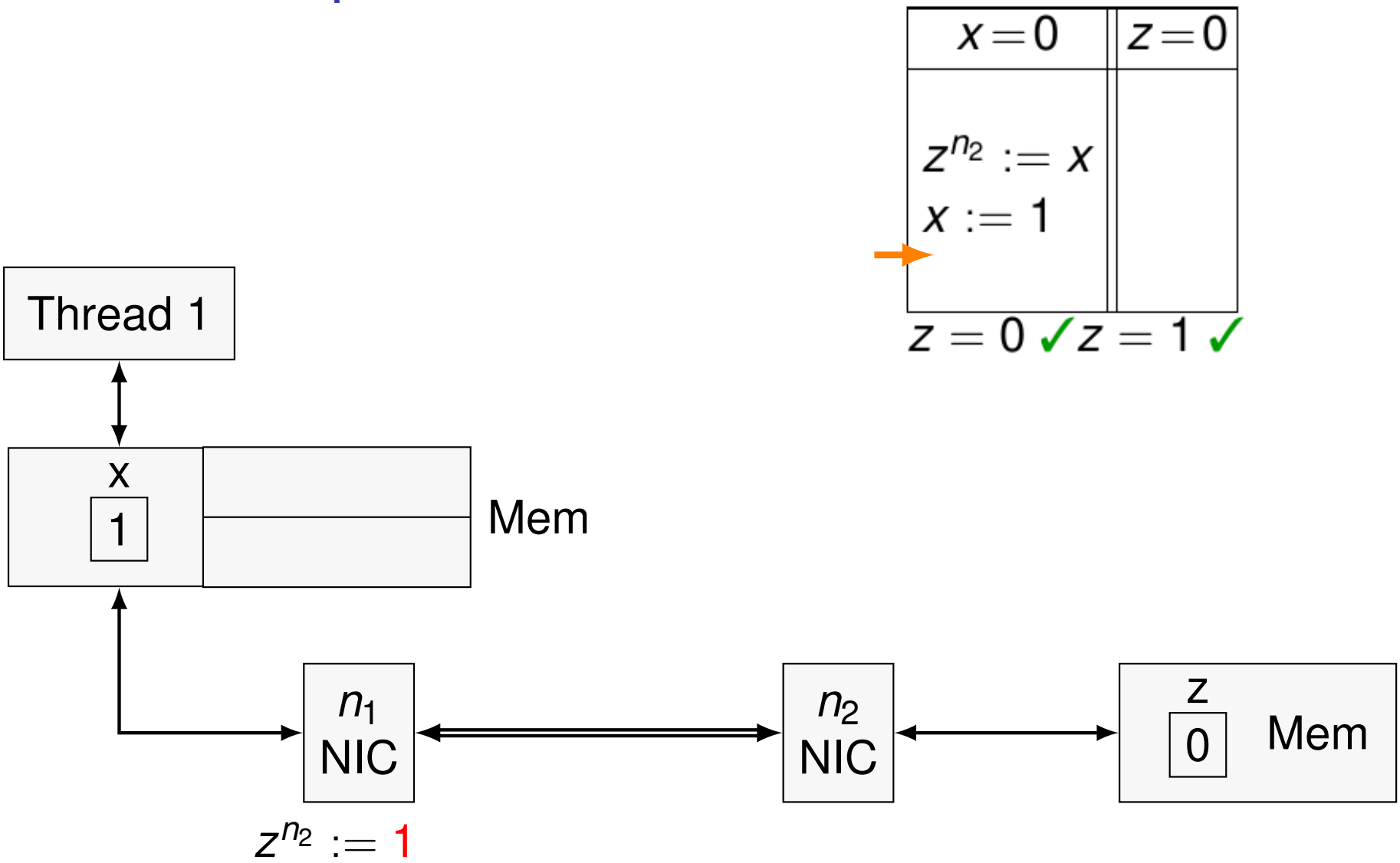




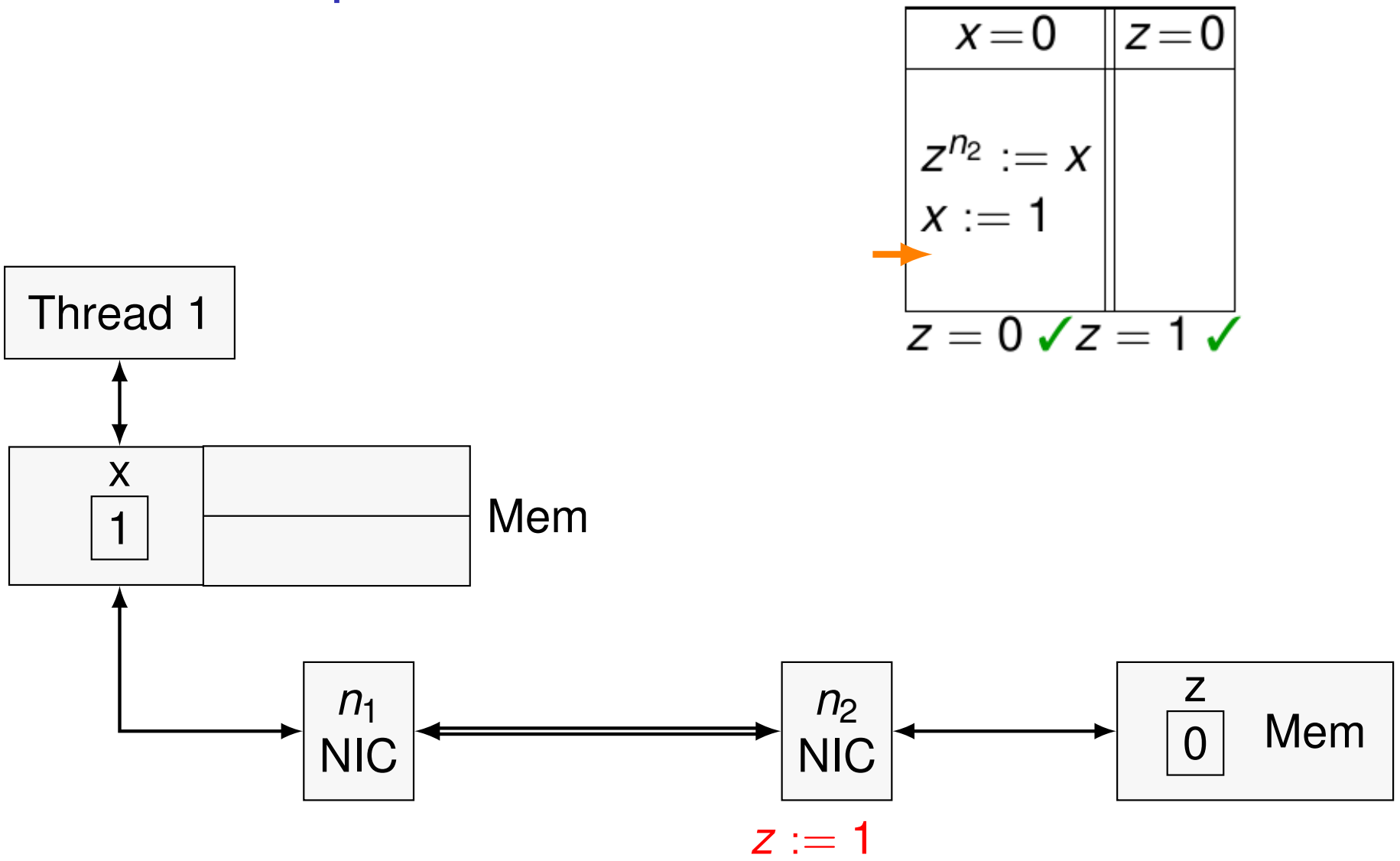
RDMA example



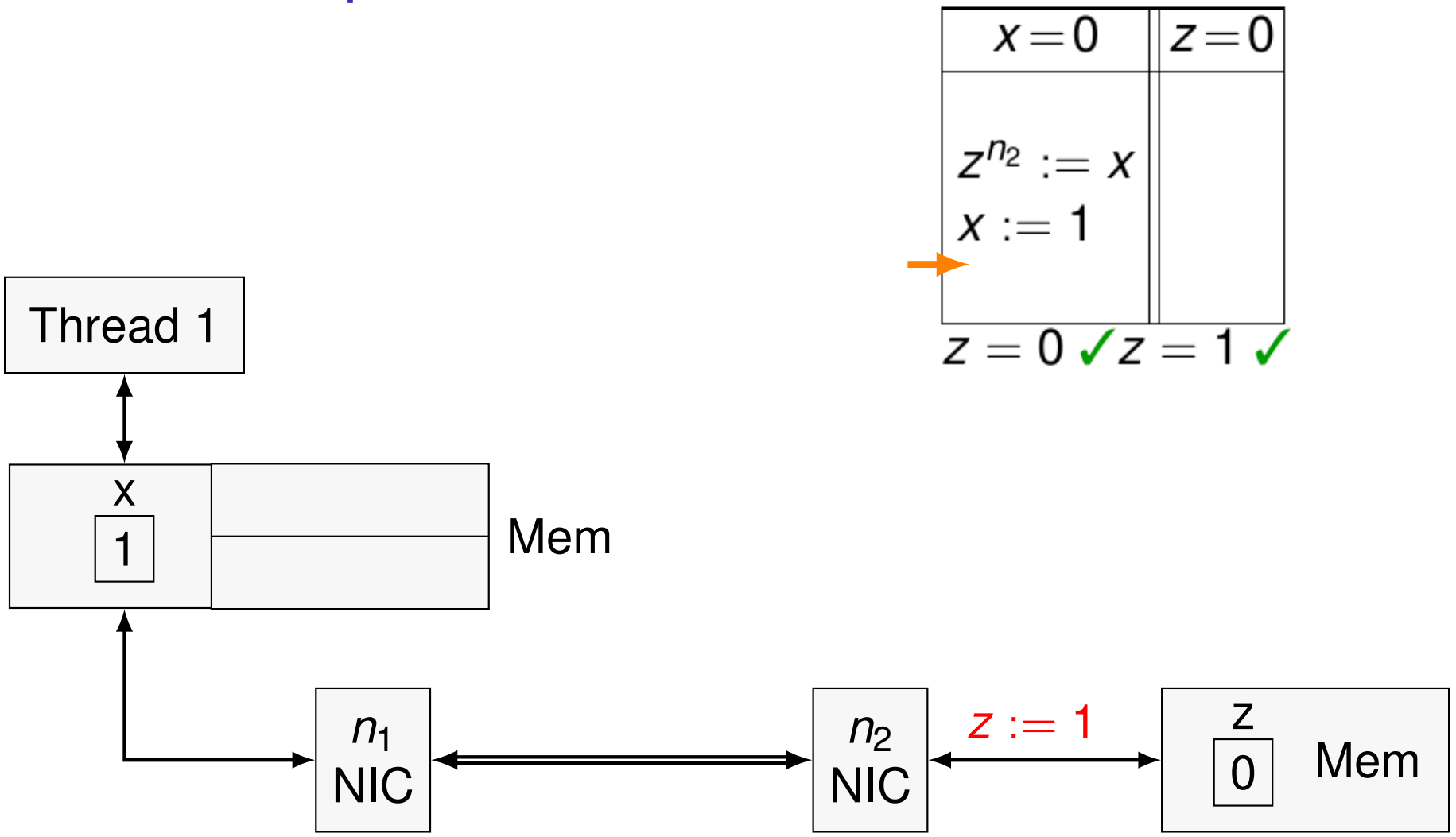
RDMA example



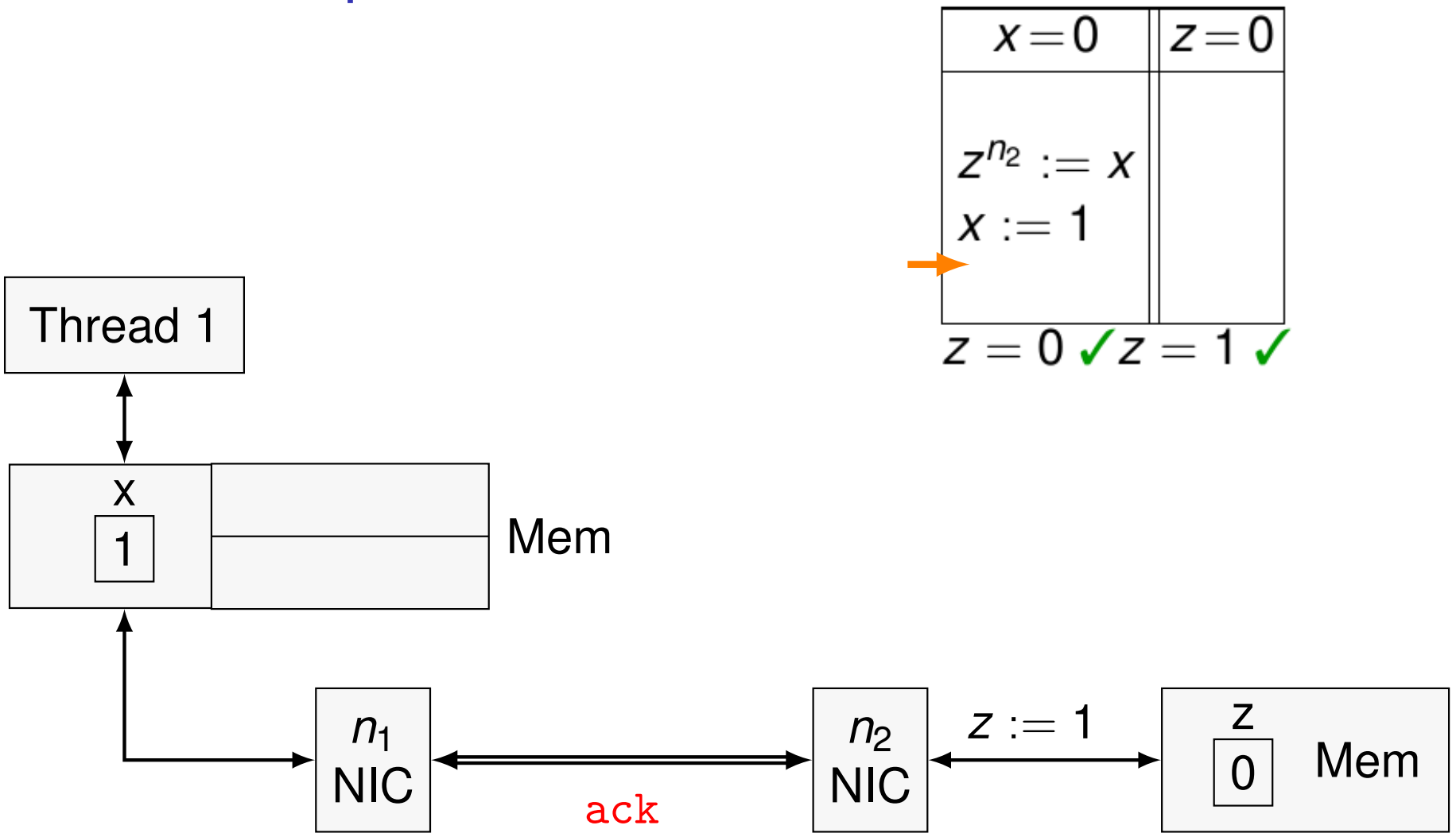
RDMA example



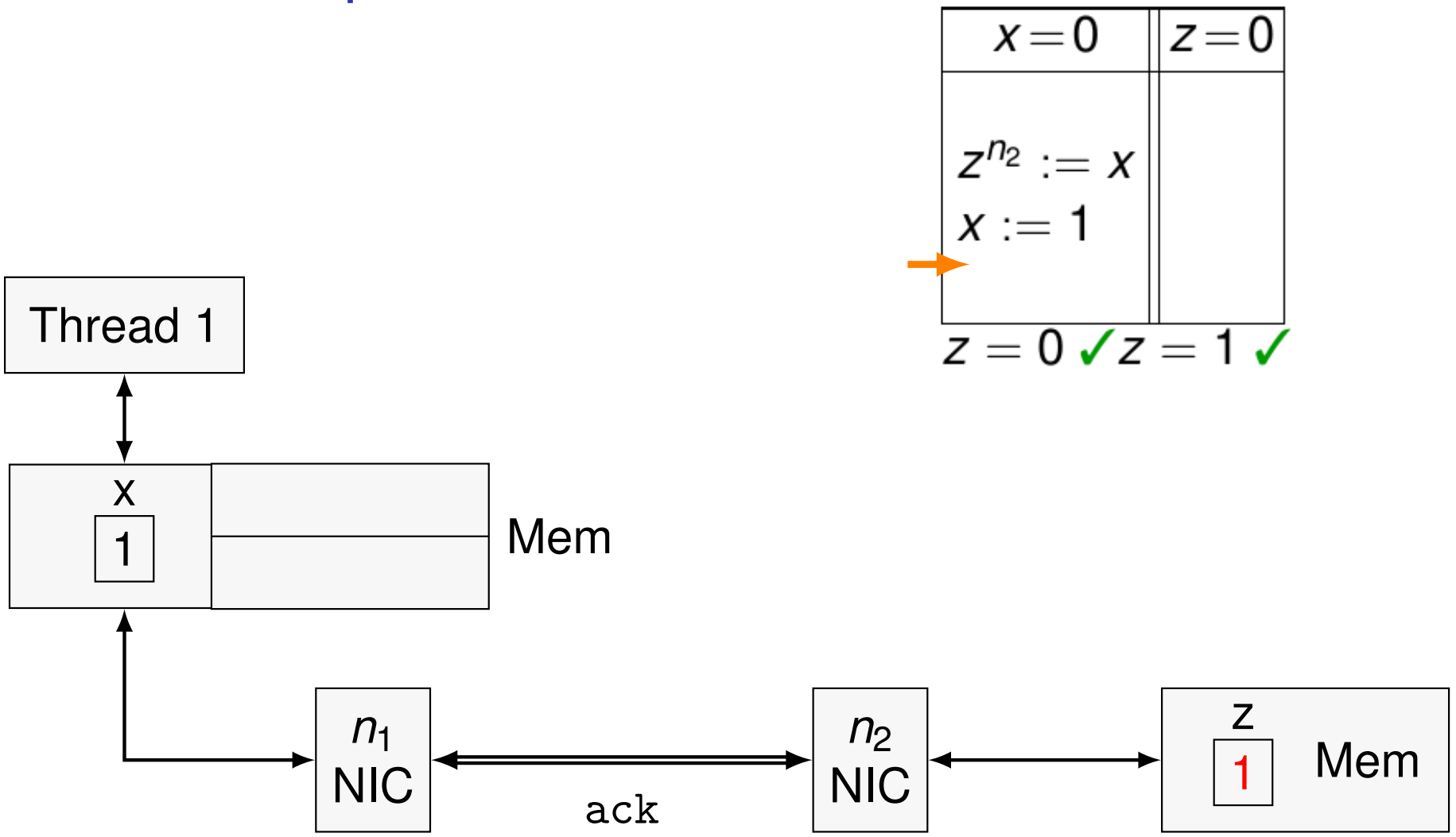
RDMA example



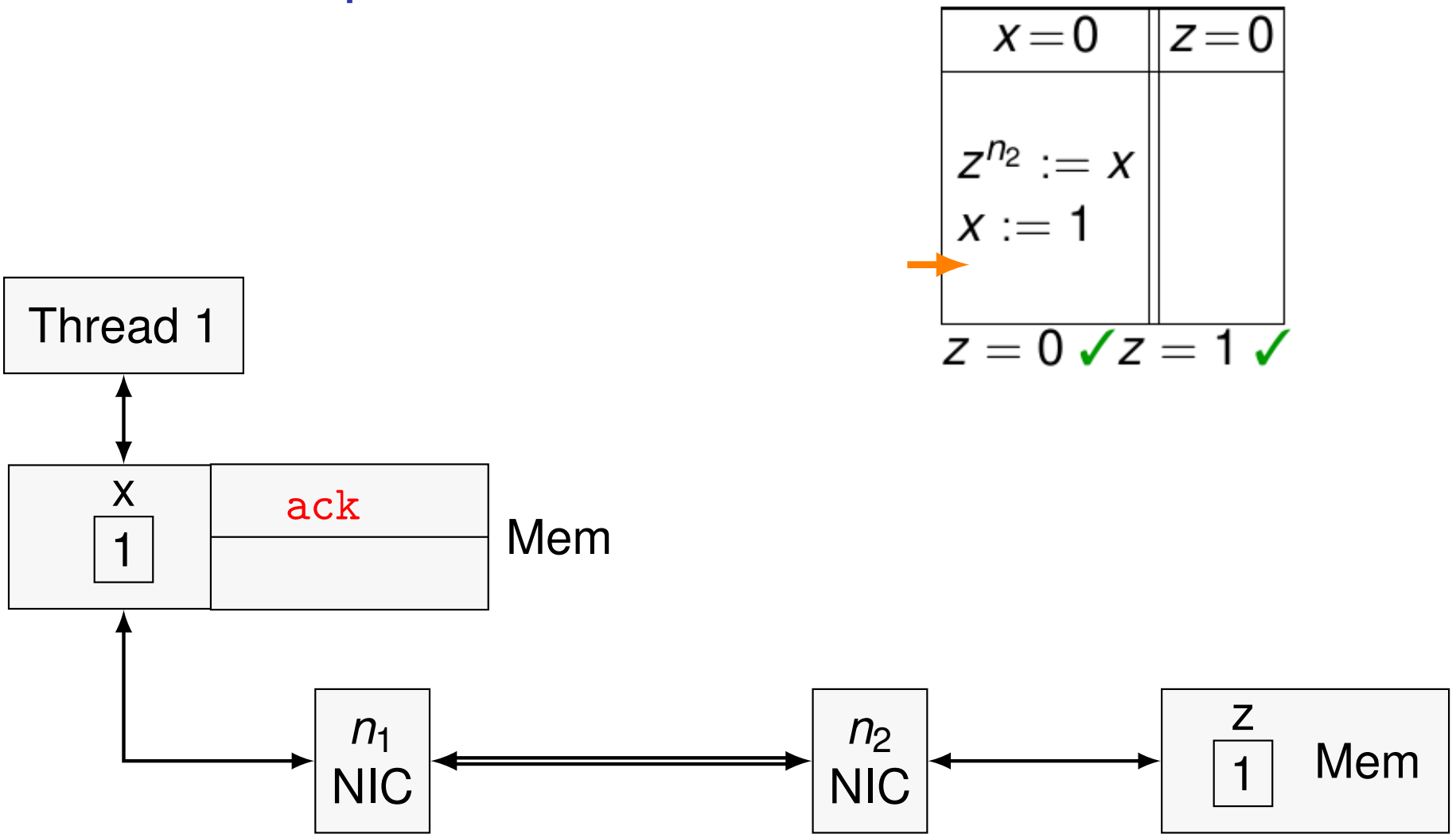
RDMA example



RDMA example



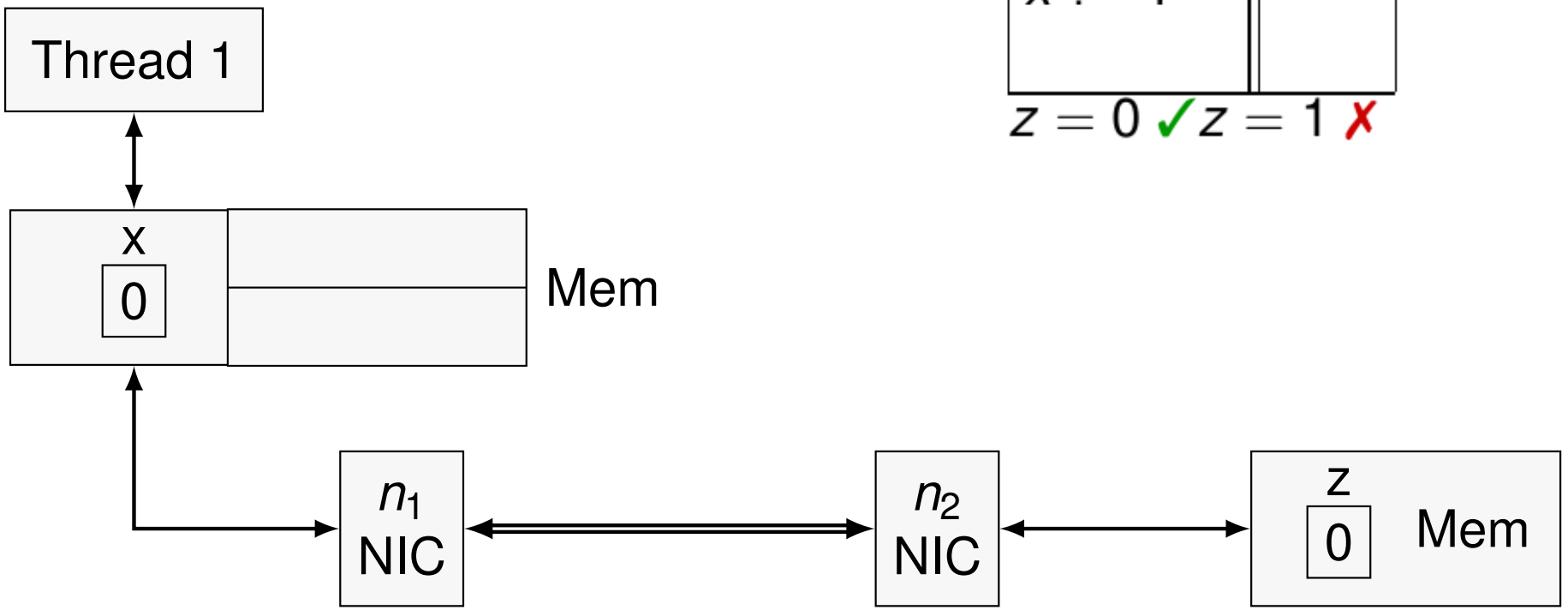
RDMA example



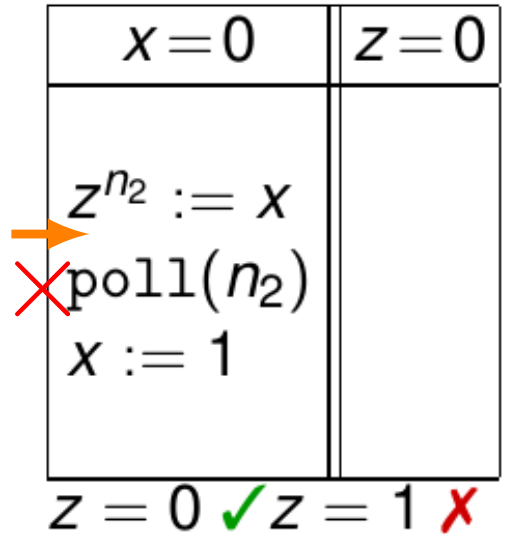
RDMA example

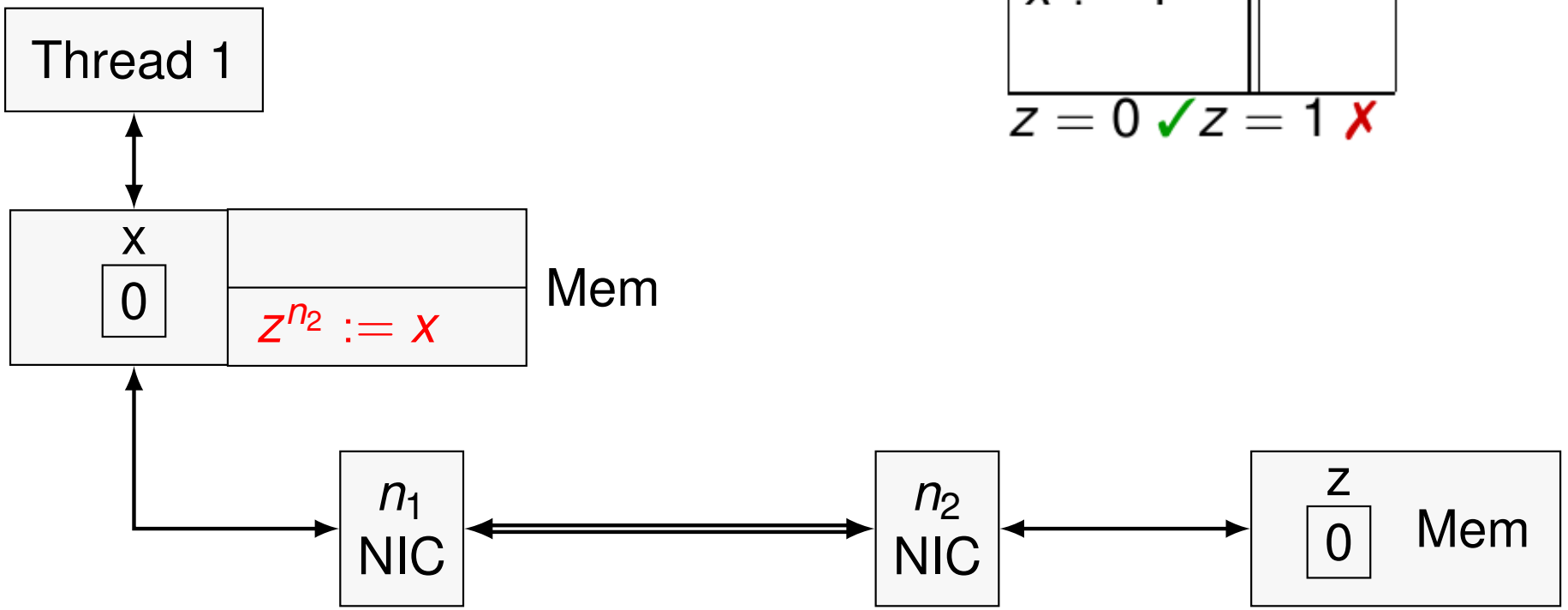
$x = 0$	$z = 0$
$z^{n_2} := x$ $\texttt{poll}(n_2)$ $x := 1$	

$z = 0$ ✓ $z = 1$ ✗

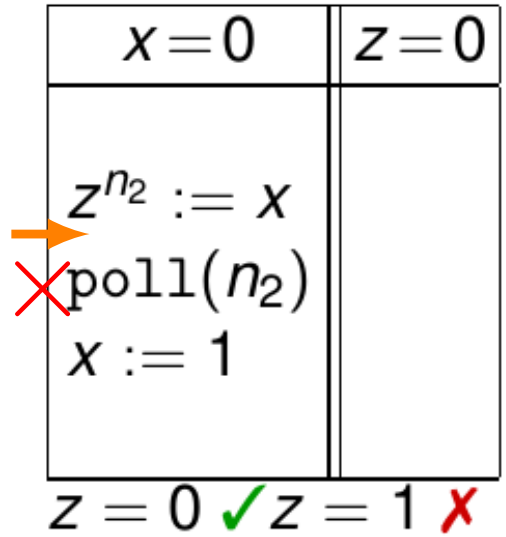


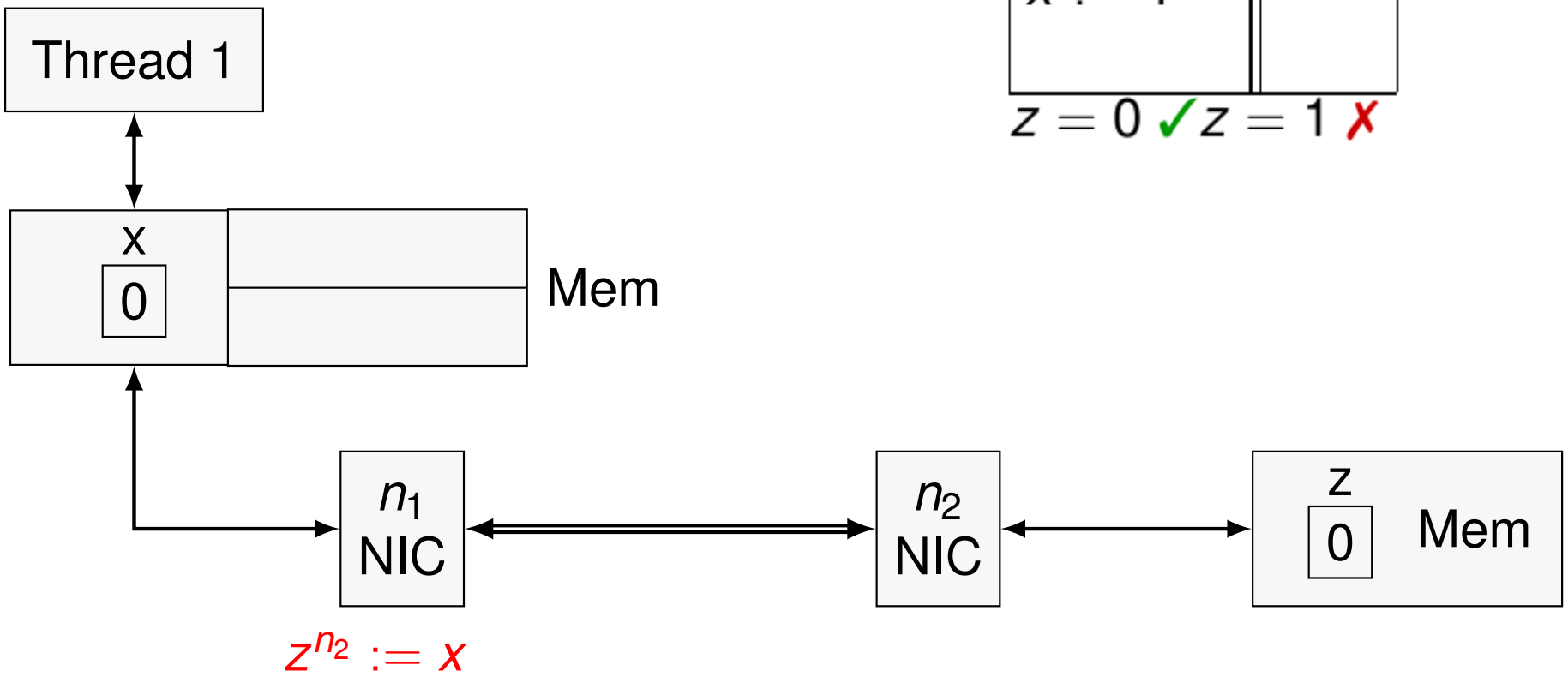
RDMA example



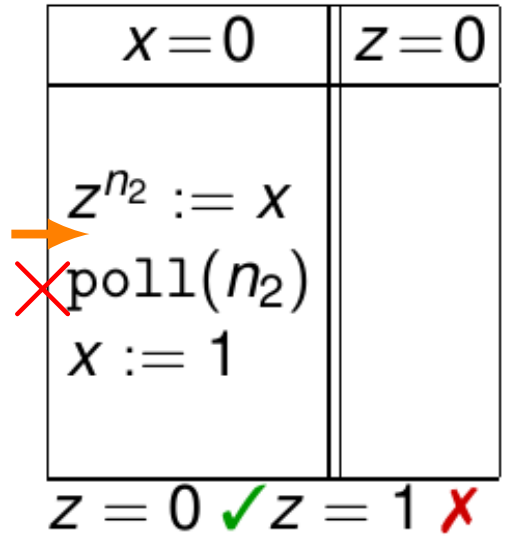


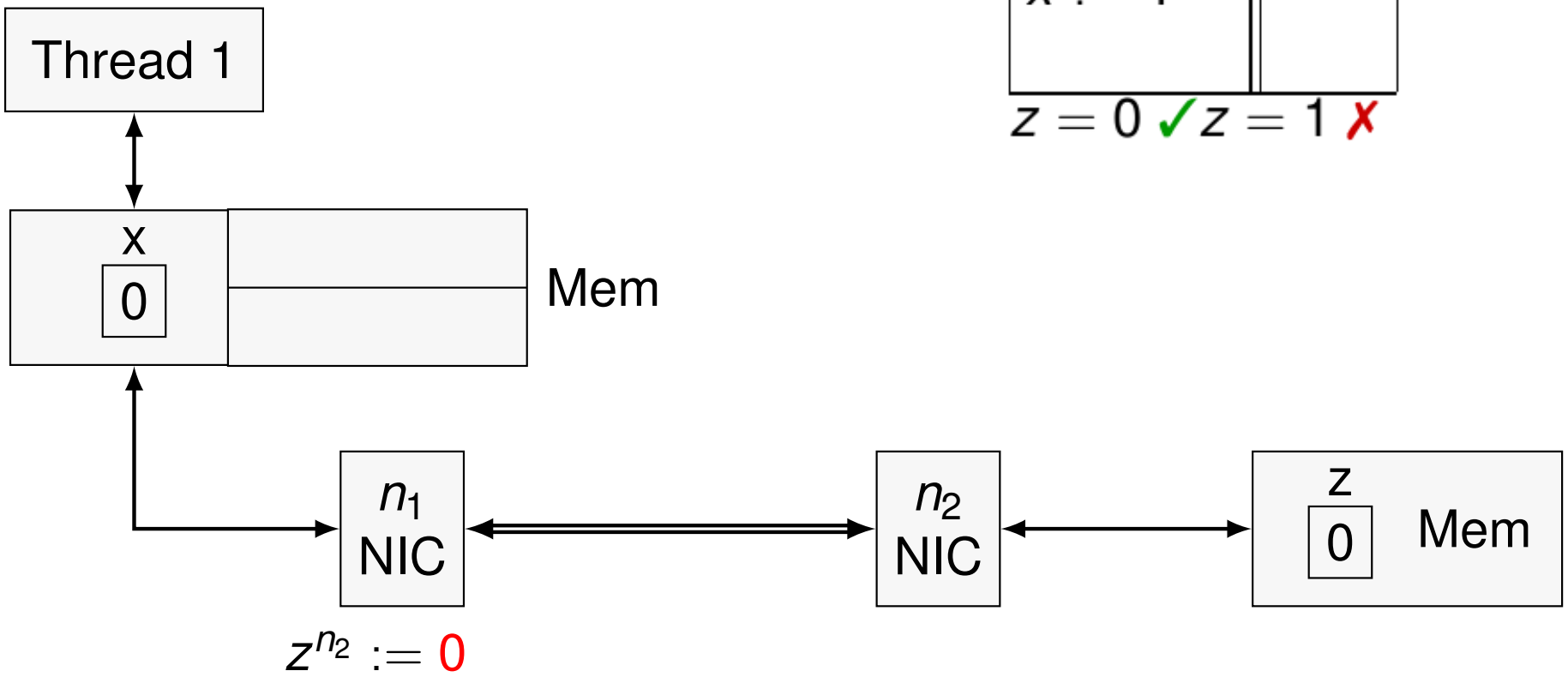
RDMA example



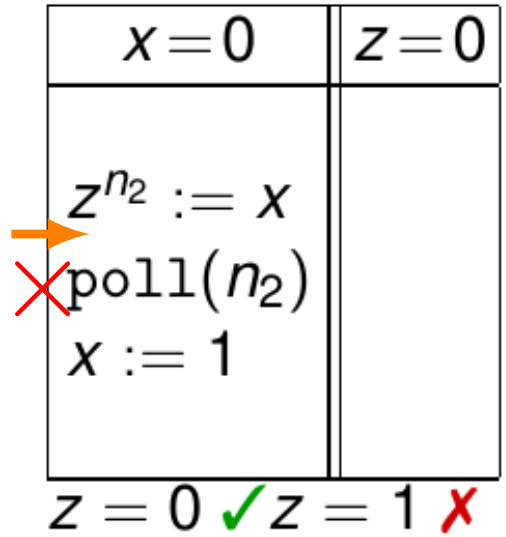


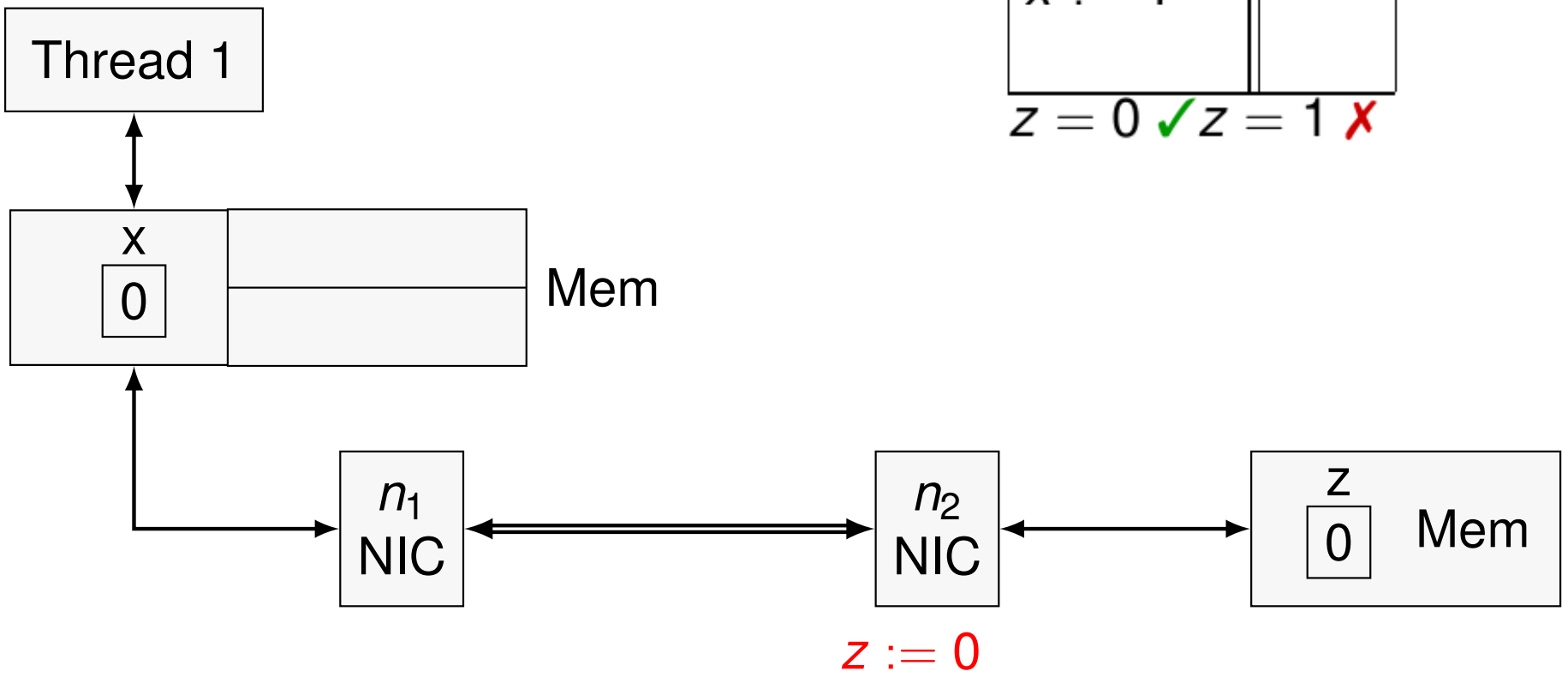
RDMA example



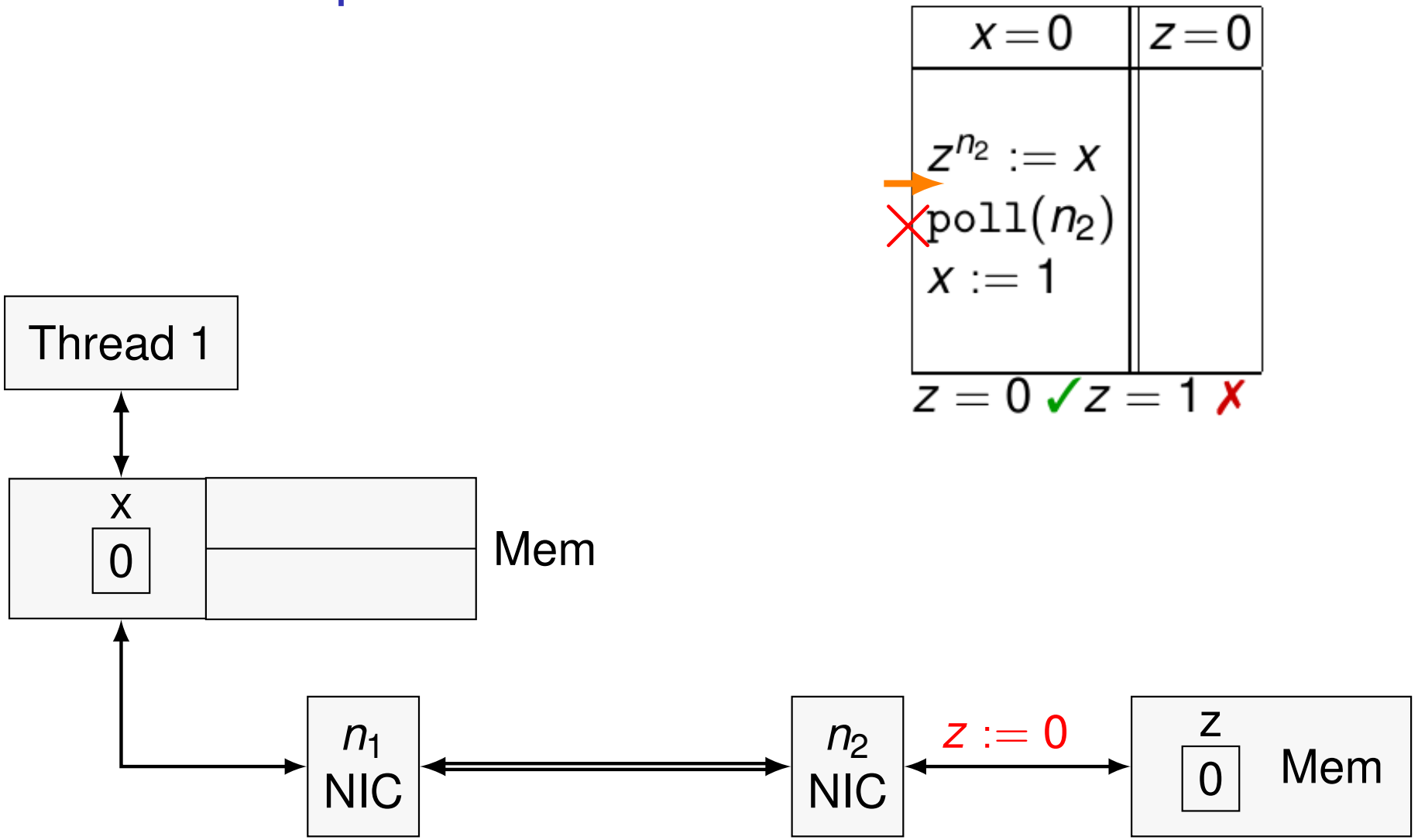


RDMA example

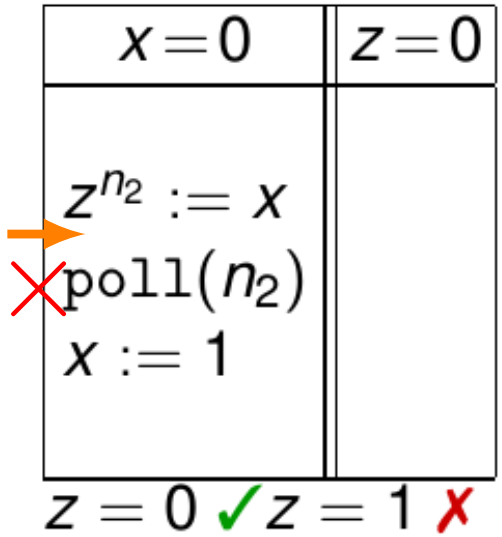


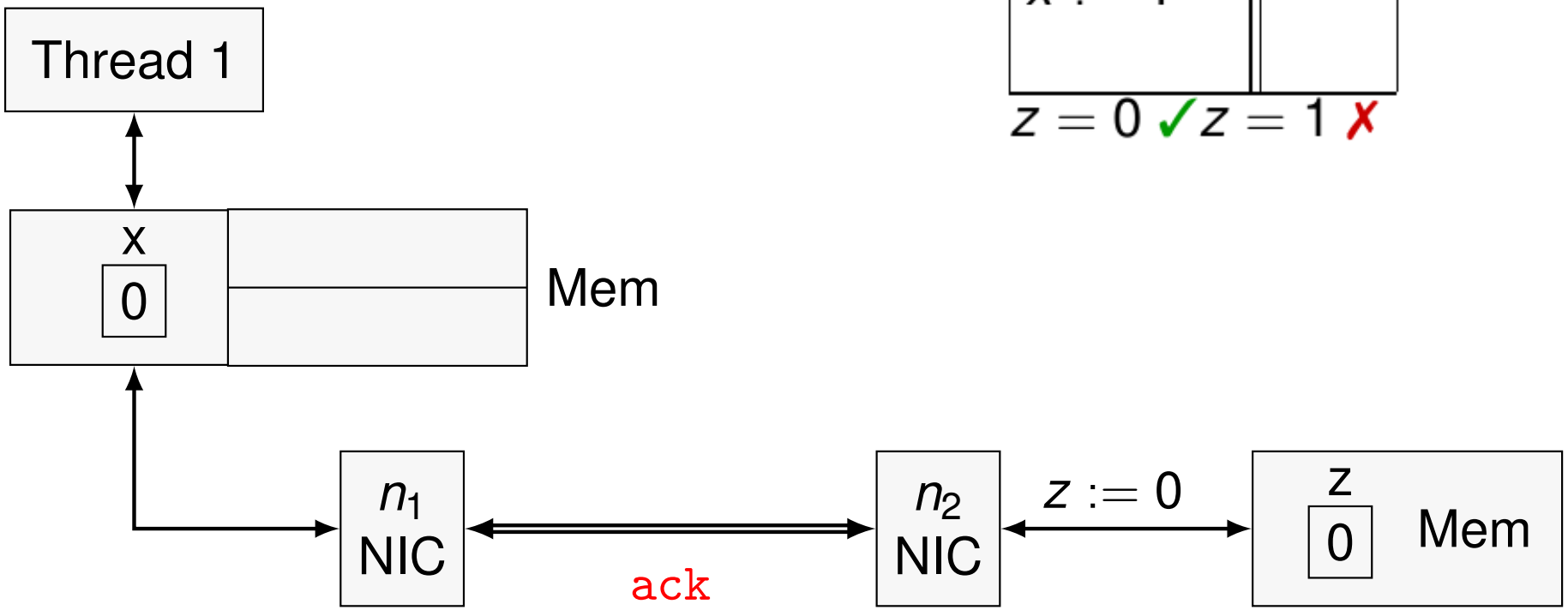


RDMA example

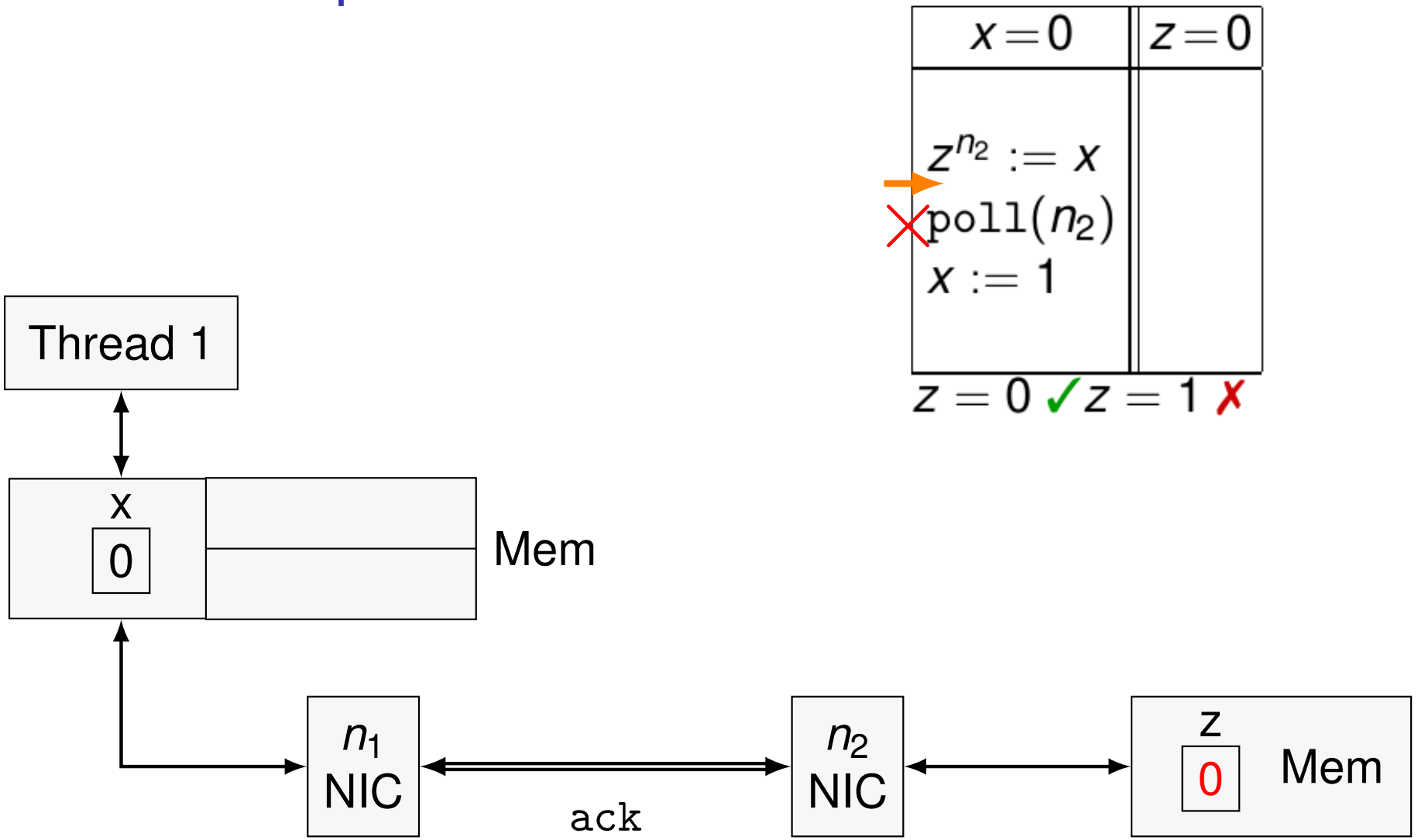


RDMA example

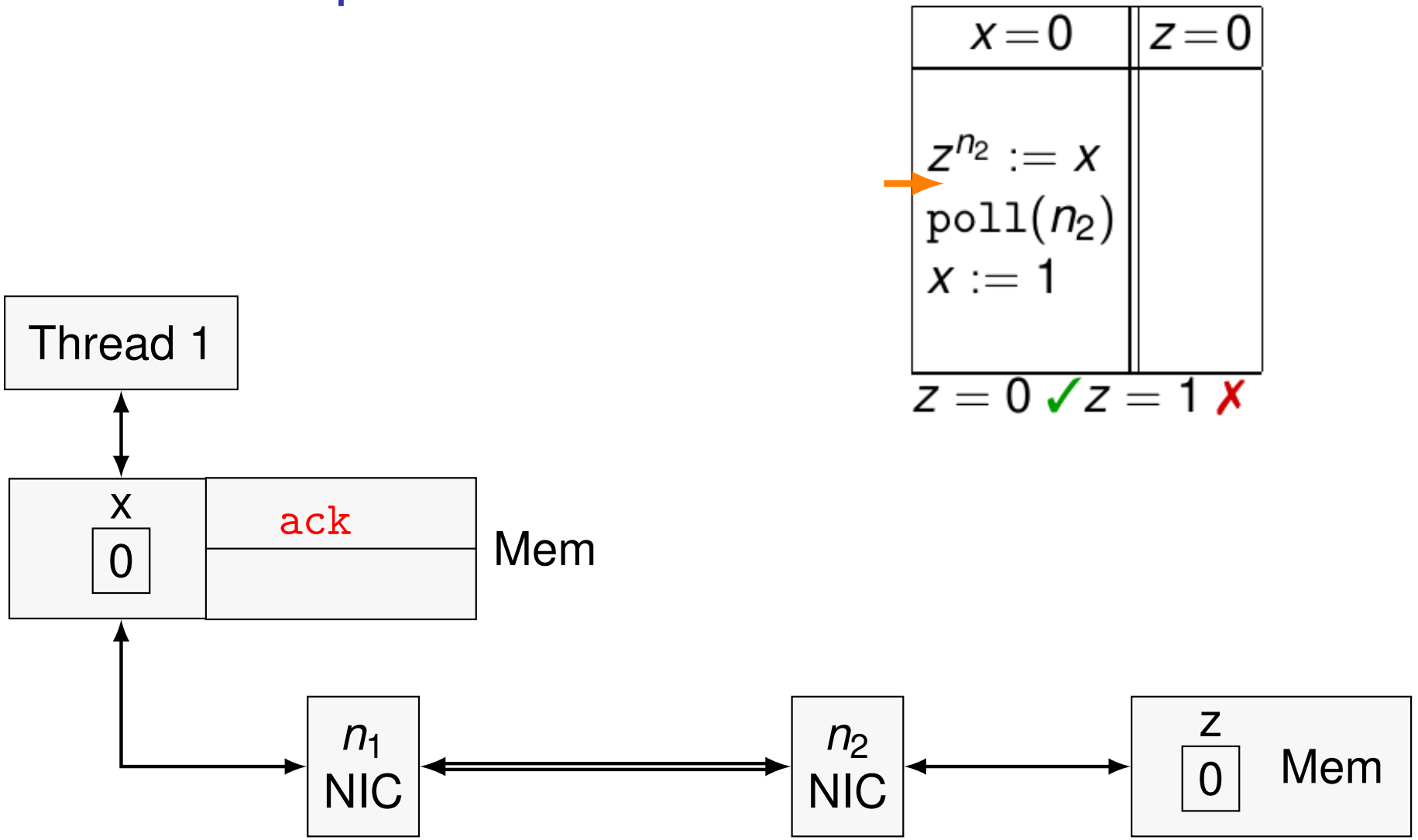




RDMA example

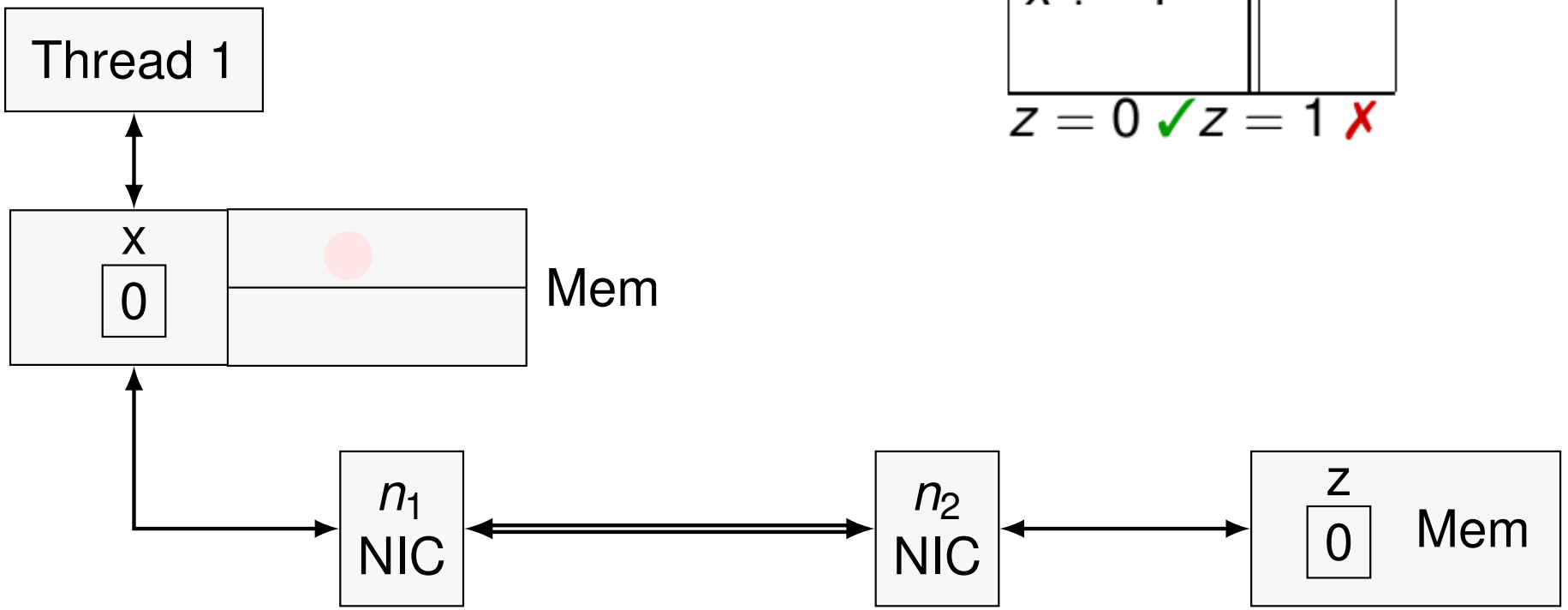


RDMA example



RDMA example

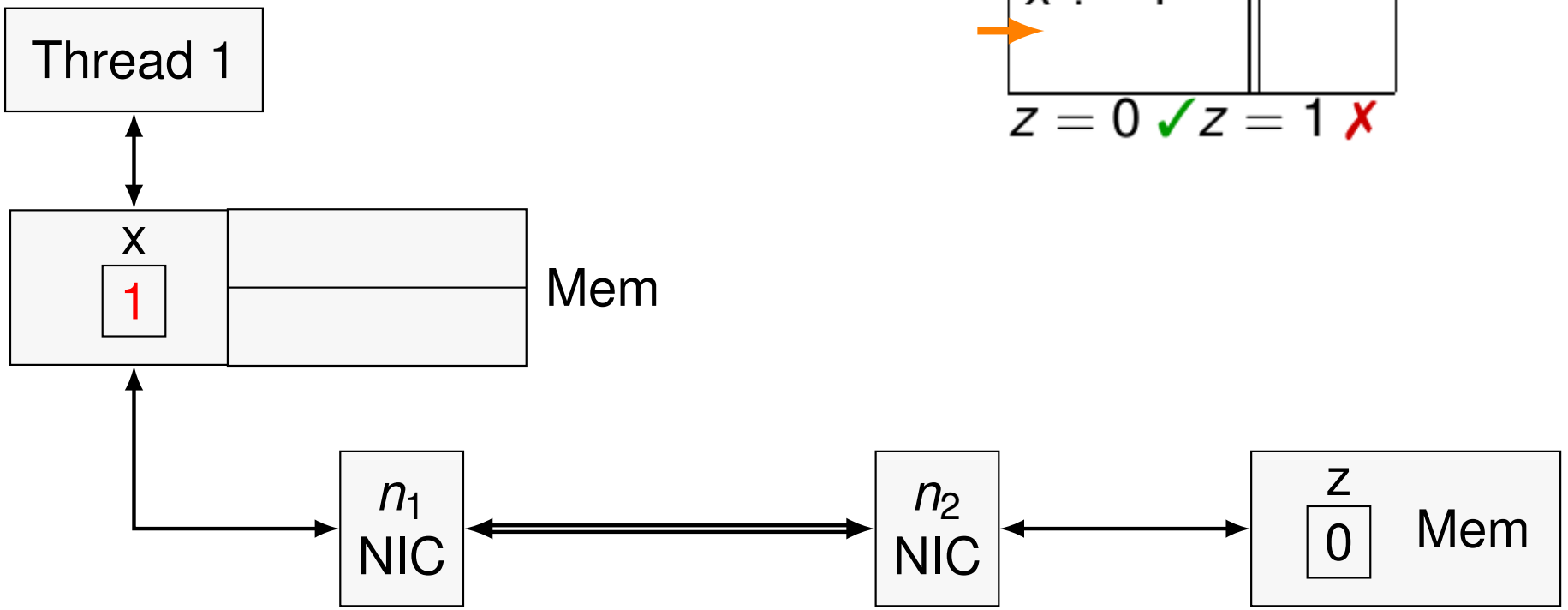
$x = 0$	$z = 0$
$z^{n_2} := x$ $\texttt{poll}(n_2)$ $x := 1$	
$z = 0$ ✓ $z = 1$ ✗	



RDMA example

$x = 0$	$z = 0$
$z^{n_2} := x$ $\texttt{poll}(n_2)$ $x := 1$	

$z = 0$ ✓ $z = 1$ ✗



DONE: formalising RDMA semantics

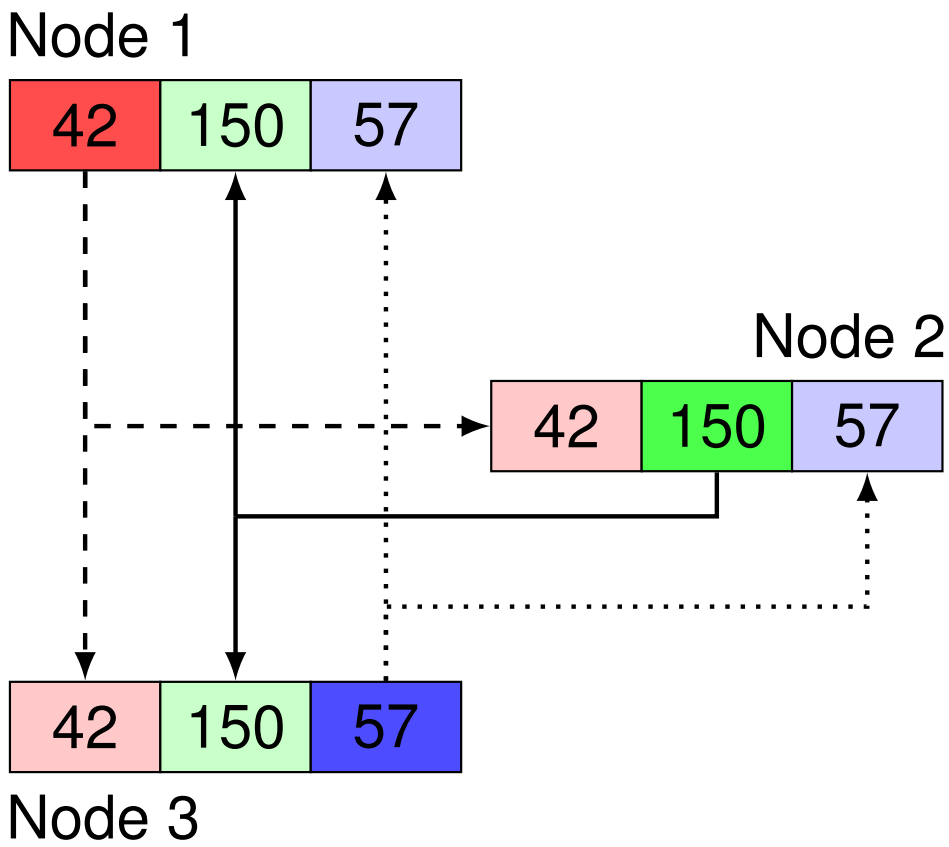
Operational Semantics Formalised in Coq:

- nodes and threads
- x86-TSO store buffers
- IB Verbs (API used by the CPU)
- internal semantics

WIP: defining shared data structures

Example:
Shared State Table (SST)[1]

Each node has a value and a copy of other nodes values

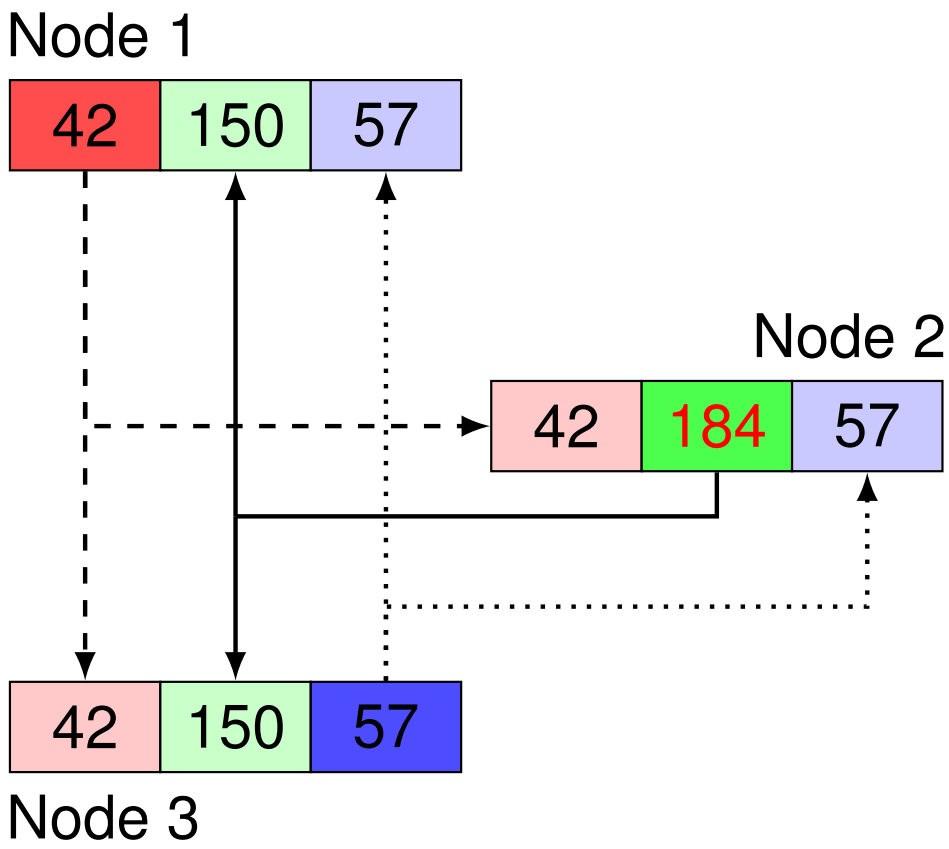


[1] Jha et al. 2018, “Derecho: Fast State Machine Replication for Cloud Services”.

WIP: defining shared data structures

Example:
Shared State Table (SST)[1]

Each node has a value and a copy of other nodes values

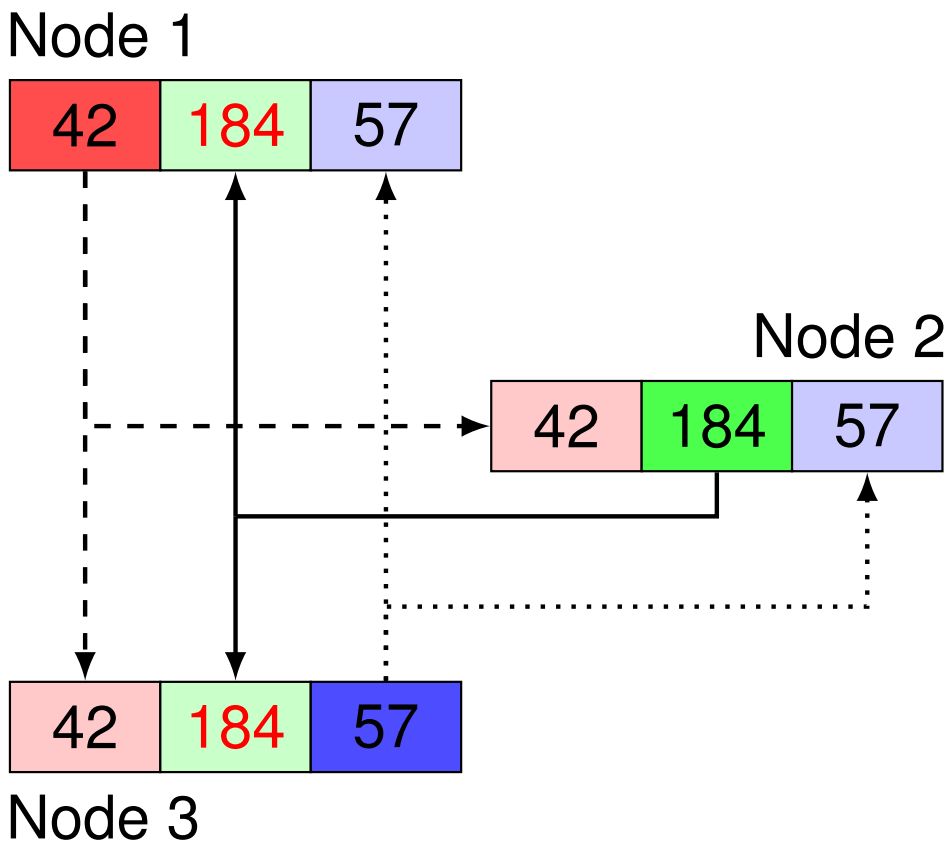


[1] Jha et al. 2018, “Derecho: Fast State Machine Replication for Cloud Services”.

WIP: defining shared data structures

Example:
Shared State Table (SST)[1]

Each node has a value and a copy of other nodes values

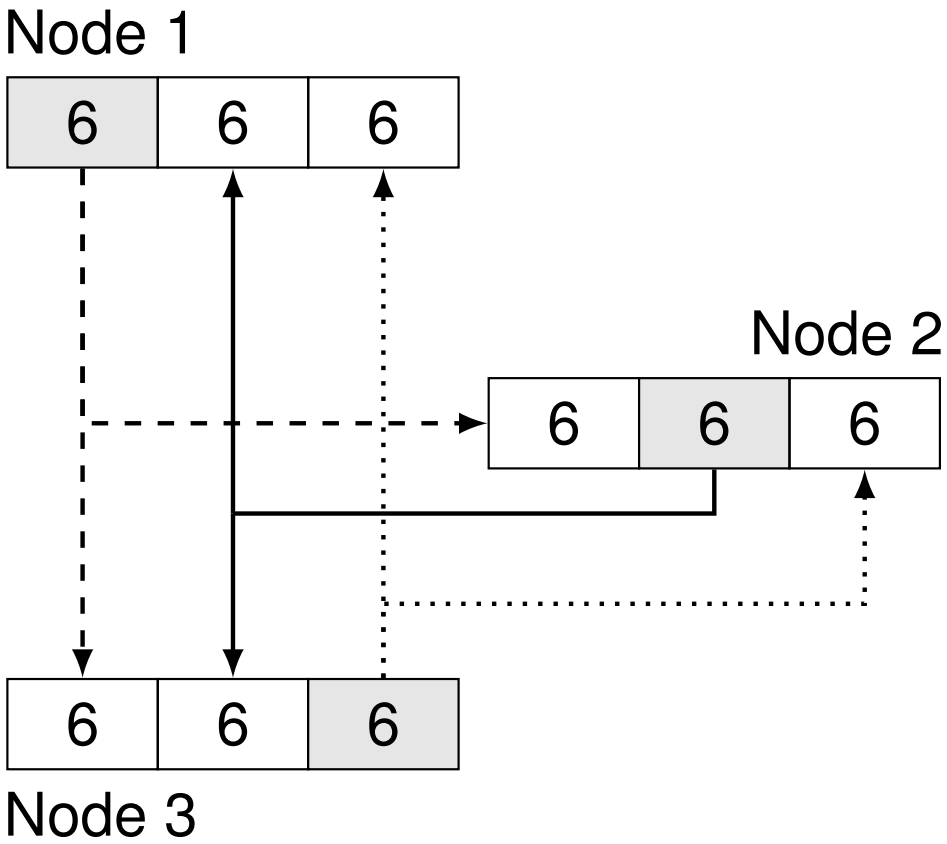


[1] Jha et al. 2018, “Derecho: Fast State Machine Replication for Cloud Services”.

FUTURE: certifying functions

Example: barrier

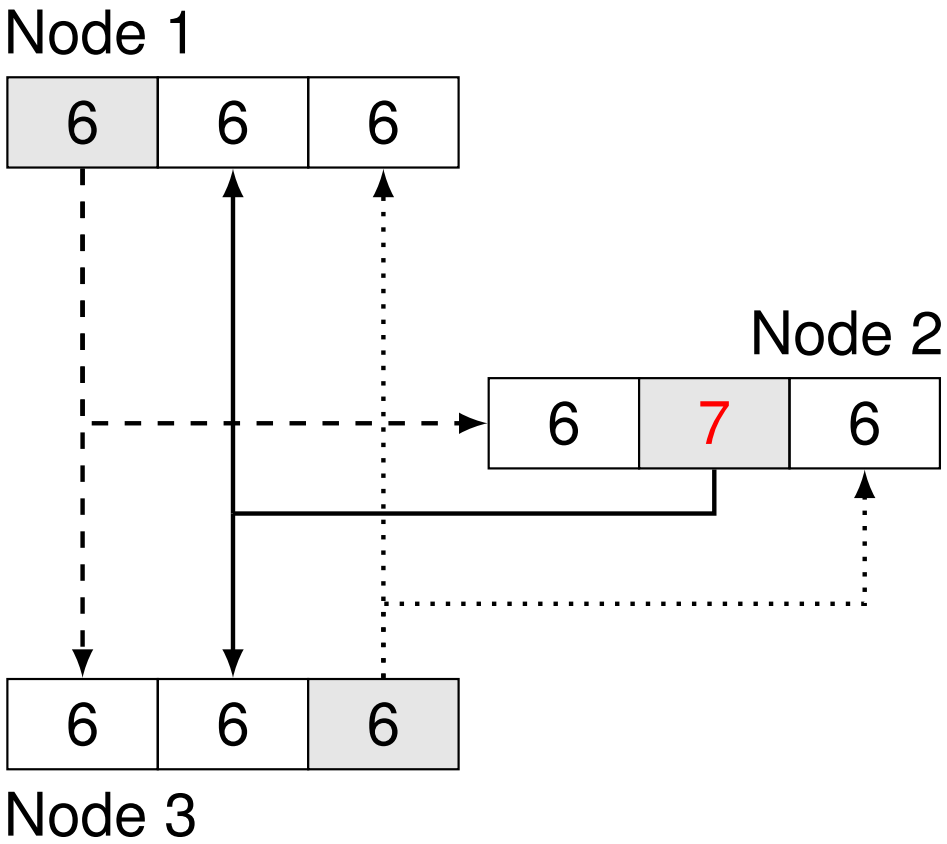
```
barrier():
1   val <- val+1
2   broadcast(val)
3   waiting <- true
4   while (waiting):
5     waiting <- false
6     for n in remotes:
7       if n.val < val:
8         waiting <- true
```



FUTURE: certifying functions

Example: barrier

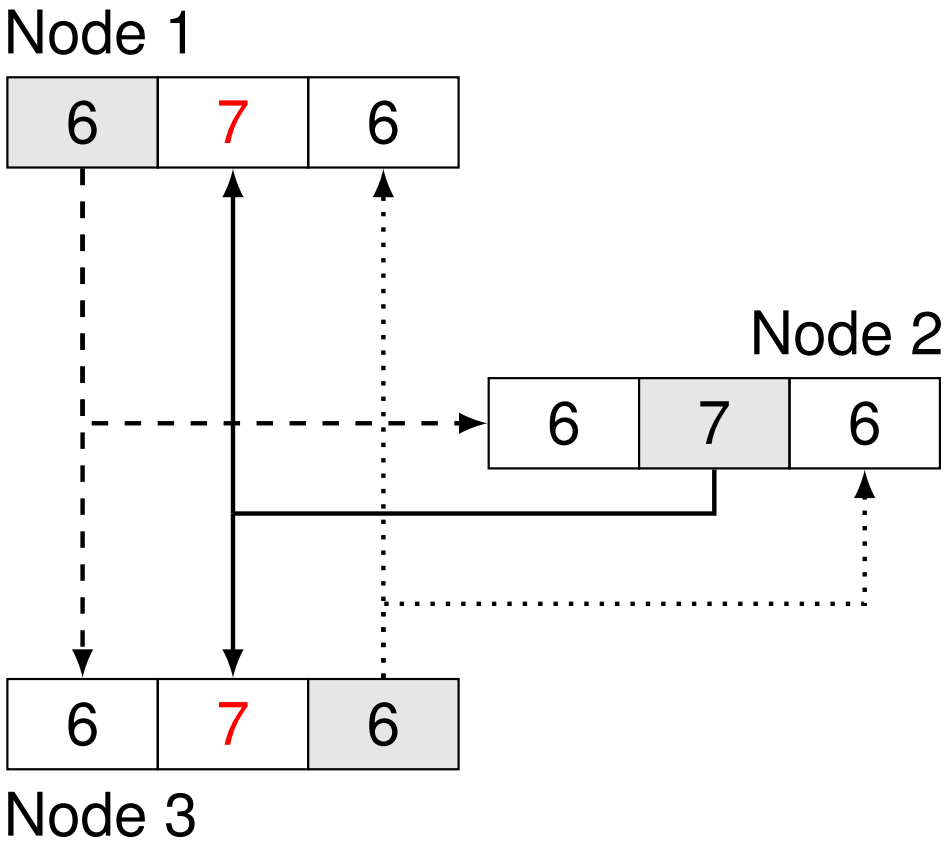
```
barrier():
1   val <- val+1
2   broadcast(val)
3   waiting <- true
4   while (waiting):
5     waiting <- false
6     for n in remotes:
7       if n.val < val:
8         waiting <- true
```



FUTURE: certifying functions

Example: barrier

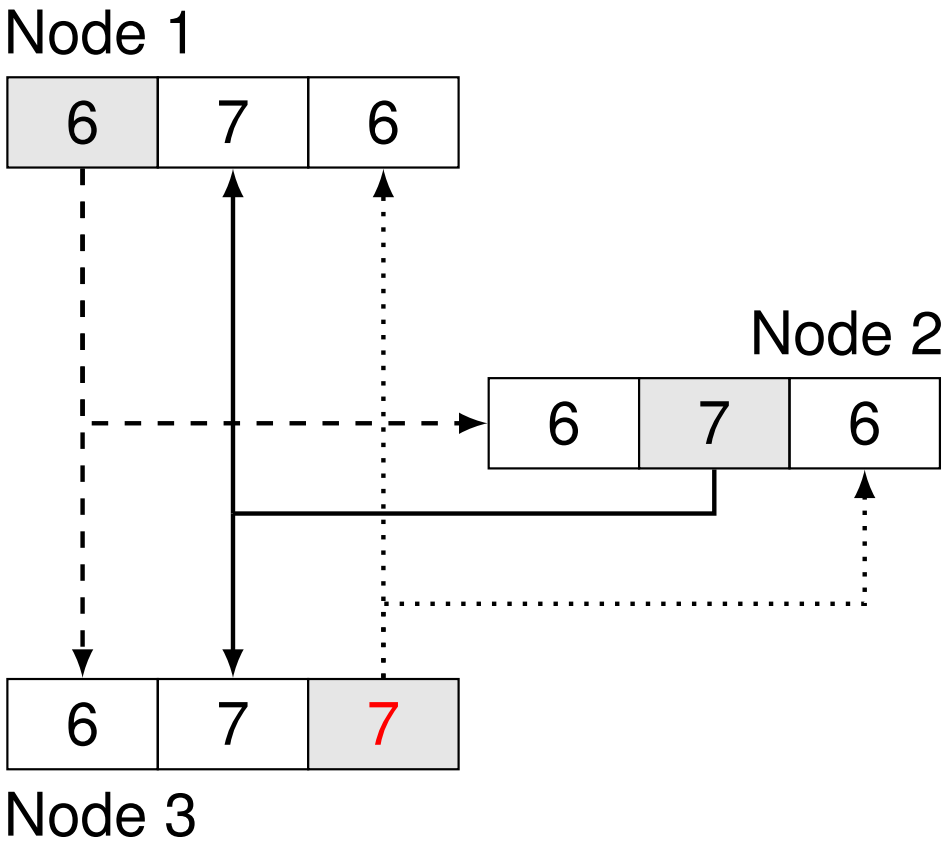
```
barrier():
1   val <- val+1
2   broadcast(val)
3   waiting <- true
4   while (waiting):
5     waiting <- false
6     for n in remotes:
7       if n.val < val:
8         waiting <- true
```



FUTURE: certifying functions

Example: barrier

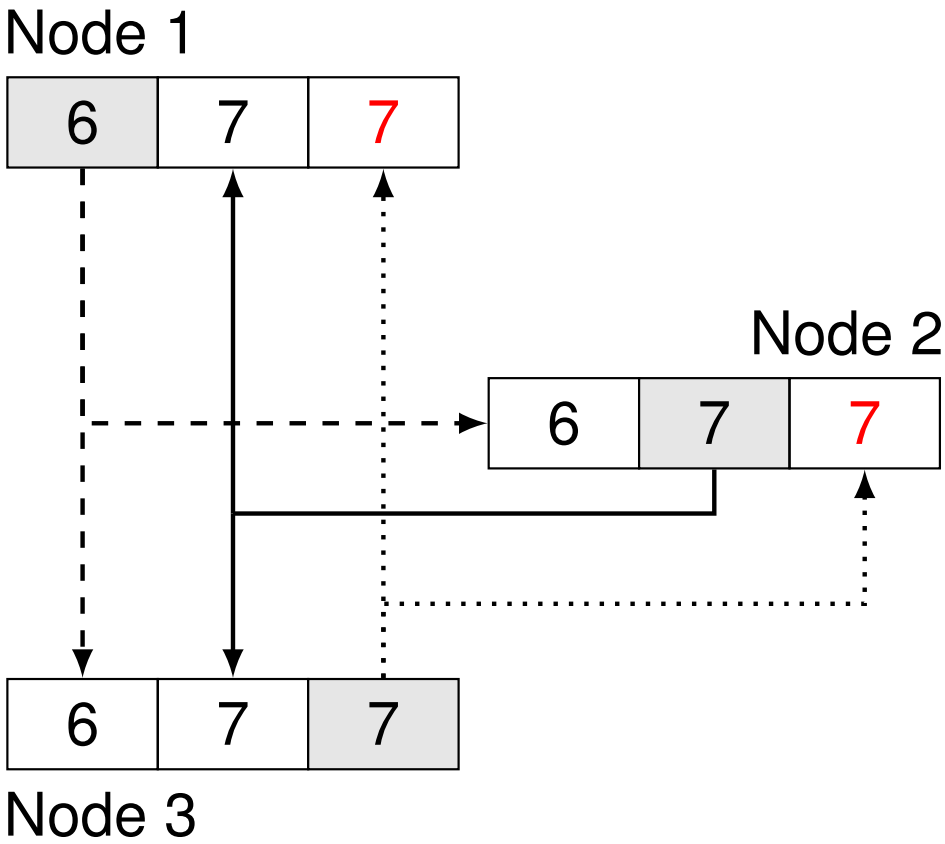
```
barrier():
1   val <- val+1
2   broadcast(val)
3   waiting <- true
4   while (waiting):
5     waiting <- false
6     for n in remotes:
7       if n.val < val:
8         waiting <- true
```



FUTURE: certifying functions

Example: barrier

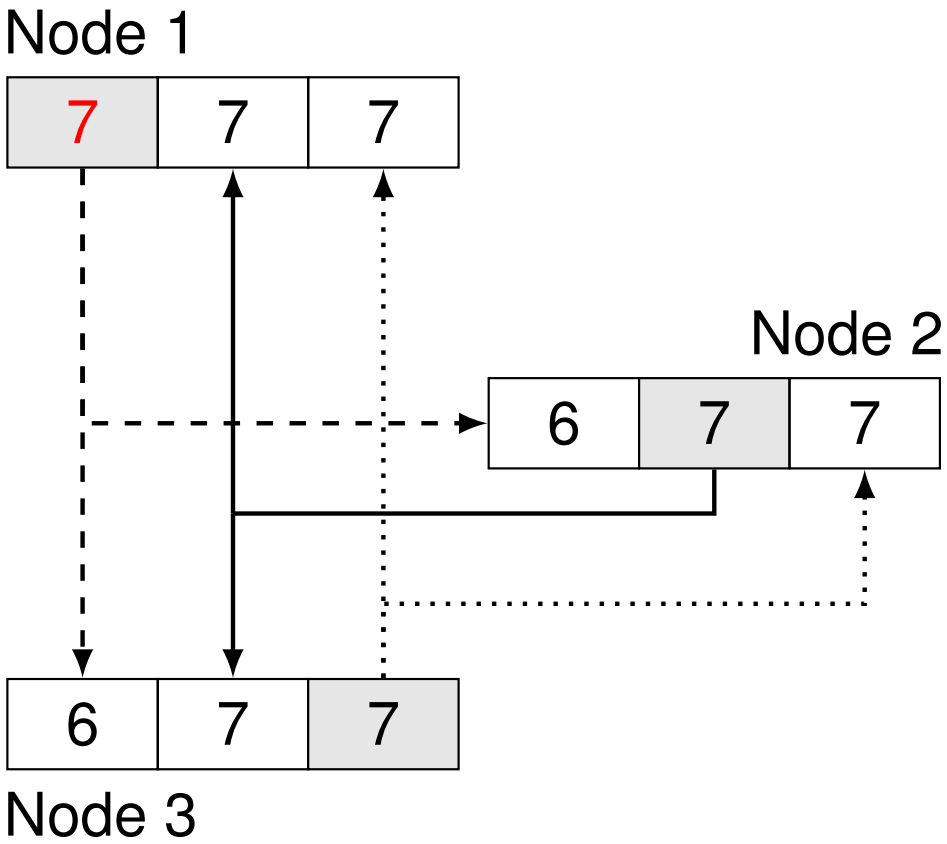
```
barrier():
1   val <- val+1
2   broadcast(val)
3   waiting <- true
4   while (waiting):
5     waiting <- false
6     for n in remotes:
7       if n.val < val:
8         waiting <- true
```



FUTURE: certifying functions

Example: barrier

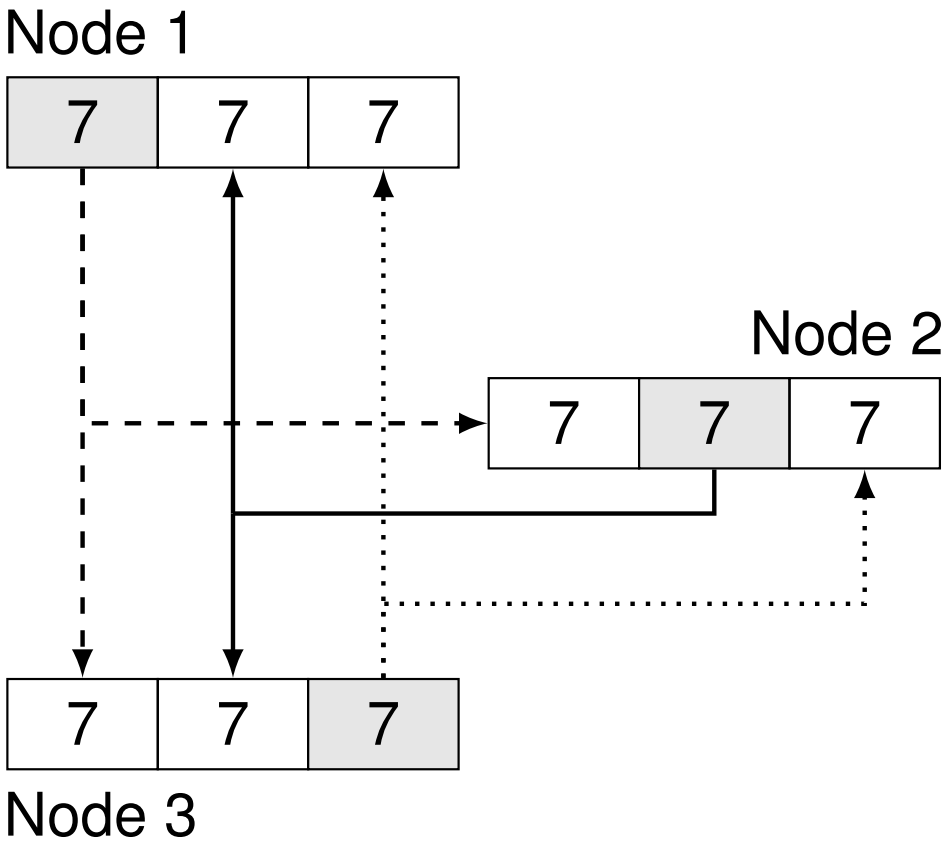
```
barrier():
1   val <- val+1
2   broadcast(val)
3   waiting <- true
4   while (waiting):
5     waiting <- false
6     for n in remotes:
7       if n.val < val:
8         waiting <- true
```



FUTURE: certifying functions

Example: barrier

```
barrier():
1   val <- val+1
2   broadcast(val)
3   waiting <- true
4   while (waiting):
5     waiting <- false
6     for n in remotes:
7       if n.val < val:
8         waiting <- true
```



Summary

Context:

- Old technology for fast data transfer with obscure semantics
- Getting cheaper and usable for more applications
- In need of support for formal verification

Our work:

- DONE: formalisation of the base RDMA semantics
- WIP: formalising shared data structures of libraries
- FUTURE: certifying functions/programs